Report: Multiple Views of ReLU Neural Networks

1. Introduction

The ReLU activation function is usually introduced as a simple nonlinearity. However, there are several equivalent but conceptually distinct ways to interpret what it is doing inside a neural network:

1. **Functional view** (standard textbook perspective)
2. **Switching / gating view** (connection-based interpretation)
3. **Decision matrix view** (linear algebra formulation)
4. **Synthetic matrix view** (input-dependent linear operator)

Each viewpoint emphasizes a different structural property, and together they provide a much deeper understanding of how deep ReLU networks operate.

---

2. Standard Functional View

A ReLU unit is defined elementwise as:

$\mathrm{ReLU}(z_i) = \max(0, z_i)$

A standard layer is written as:

$y = \mathrm{ReLU}(Wx)$

Interpretation

- Compute a weighted sum (Wx)
- Apply a nonlinear function that:
    - Keeps positive values
    - Zeros out negative values

Key properties

- Nonlinearity is explicit
- Easy to analyze locally
- Widely used in optimization and gradient-based training

Limitation of this view

- It hides the **combinatorial structure** created by many ReLU decisions
- It does not emphasize that the network is **piecewise linear**

---

3. Switching / Gating View

In this view, each ReLU is interpreted as a **binary switch**:

- If $z_i > 0$: connection is **ON**
- If $z_i \leq 0$: connection is **OFF**

So instead of thinking "apply a function," we think:

> "Allow this signal to pass, or block it."

Layer reinterpretation

1. Compute $z = Wx$
2. For each neuron:
    - If active → pass z_i
    - If inactive → output 0

This is equivalent to inserting switches after each neuron.

Key Insight

A ReLU layer is not just transforming values — it is **selecting a subnetwork**.

- Active neurons form a **live computation path**
- Inactive neurons are effectively **removed**

So the network dynamically constructs a computation graph depending on the input.

Composition Effect

In deep networks:

- Each layer makes switching decisions
- These decisions:
    - Enable certain weighted sums
    - Disable others

This creates:

> **A vast combinatorial space of possible active linear pathways**

Each input selects one such pathway.

---

4. Decision Matrix View

We now formalize the switching behavior.

Define a diagonal matrix $D(x)$:

- $D_{ii}(x) = 1$ if neuron i is active
- $D_{ii}(x) = 0$ if neuron i is inactive

Then:

$ReLU(Wx) = DWx$

Interpretation

- $D(x)$ encodes **all switching decisions**
- It is:
    - Input-dependent
    - Sparse
    - Binary (0/1)

Why this matters

This converts a nonlinear layer into:

> **A linear transformation conditioned on input**

So:

- Nonlinearity is now moved into **which matrix is selected**
- The computation itself is linear once $D(x)$ is fixed

---

5. Synthetic Matrix View

Define the **synthetic matrix**:

$$S(x) = WD(x)$$

Then the layer becomes:

$$y = S(x)x$$

Interpretation

- The layer is a **linear operator**
- But the operator depends on x

So instead of:

> "apply a fixed matrix + nonlinearity"

we have:

> "select a matrix based on input, then apply it"

Key Insight

A ReLU layer behaves like a **lookup system over matrices**:

- Input determines $D(x)$
- $D(x)$ determines $S(x)$
- $S(x)$ performs the transformation

---

6. Deep Network Composition

Consider a deep network:

$y = S_L...S_2S_1Wx$

Then the entire deep network is:

> **A single linear transformation chosen from a large family**

Structure of this family

- Each layer contributes a binary decision pattern
- Total number of possible matrices grows combinatorially

This yields:

> **An exponential number of possible linear maps**

---

7. Comparison of the Views

View	What it emphasizes	Key object	Insight
Functional	Nonlinearity	max(0, z)	Simple computation
Switching	Connectivity	ON/OFF gates	Dynamic subnetworks
Decision matrix	Algebraic structure	D(x)	Linearization via masking
Synthetic matrix	Global behavior	S(x)	Input-dependent linear operator

---

8. Unifying Perspective

All views describe the same process:

1. Compute weighted sums
2. Decide which components survive
3. Apply only the surviving components

But they differ in *where the emphasis is placed*:

- Functional → **value transformation**
- Switching → **information flow**
- Decision matrix → **structure of activation**
- Synthetic matrix → **global linearization**

---

9. Conceptual Takeaways

1. ReLU networks are piecewise linear

- Each region of input space corresponds to a fixed D(x)
- Within that region, the network is purely linear

---

2. Nonlinearity comes from switching, not curvature

- There is no smooth bending
- Instead:
    - The model **jumps between linear regimes**

---

3. Depth builds combinatorial expressivity

- Each layer adds new switches
- This creates a massive space of possible linear maps

---

4. The network is a conditional linear system

A deep ReLU network can be viewed as:

> **A system that selects and composes linear transformations based on input-dependent binary decisions**

---

10. Closing Summary

- The **functional view** is simplest but hides structure
- The **switching view** reveals dynamic connectivity
- The **decision matrix view** exposes algebraic structure
- The **synthetic matrix view** shows the network as a conditional linear operator

Together, they show that a ReLU network is not just a nonlinear function, but:

> **A mechanism for constructing input-dependent compositions of weighted sums, governed by a hierarchy of binary decisions.**